Zapis ovir

Obvezna naloga

Mednarodni standard ISO-1234 za zapis pozicij ovir na kolesarskih poteh je takšen.

- Vsaka vrstica datoteke se nanaša na eno "vrstico" kolesarske poti.
- Prva številka v vrstici je koordinata y, zapisana na 3 mesta, z vodilnimi ničlami. Koordinata 42 je torej zapisana kot `042`.
- Sledi dvopičje.
- Sledijo pari; začetni in končni x sta ločena z `-`. Zapisana sta na 4 mesta, pri čemer je začetna koordinata poravnana desno, končna pa levo.

Če komu zveni zapleteno, živi v zmoti. Preprosto je: ovire

```
{4: [(5, 6), (9, 11)],
 5: [(9, 11), (19, 20), (30, 34)],
 13: [(5, 8), (9, 11), (17, 19), (22, 25), (90, 100)]}
```

zapišemo kot

```
004:   5-6      9-11
005:   9-11    19-20    30-34
013:   5-8      9-11    17-19    22-25    90-100
```

Napiši funkcijo `zapisi_ovire(ime_datoteke, ovire)`, ki v datoteko s podanim imenom shrani podane ovire v takšni obliki. Ovire so tokrat podane v obliki slovarja: ključi so številke vrstic, pripadajoče vrednosti pa seznami parov začetkov in koncev ovir.

Testi bodo funkciji vsakič podali drugačno ime datoteke (konkretno: trenutni čas). Če se test izvede uspešno, bodo datoteko pobrisali, sicer jo bodo pustili (in se bodo začele nabirati), tako da lahko vidite, kaj je funkcija (napačno) zapisala vanjo.

Rešitev

Odpremo datoteko za pisanje. Nato gremo čez pare koordinat `y` in seznamov začetkov in koncev. V datoteko zapišemo `y` na `3` mesta, poravnano na desno, pri čemer ne poravnavamo s presledki temveč z `0`; temu se reče `f{y:0>3}`. Znak za poravnavo na desno `>` je potreben, ker brez njega ne moremo podati znaka za poravnavo, `0`.

Nato gremo čez seznam začetkov in koncev ovir (shranili ju bomo v `x0` in `x1`) ter jih zapisujemo: `x0` preprosto zapišemo na 4 mesta, `x1` pa zapišemo na štiri mesta s poravnavo na levo. Mednju postavimo `-`. Temu se reče `f"{x0:4}-{x1:<4}"`.

Na koncu vsake vrstice gremo v novo vrstico, `"\n"`.

```python
def zapisi_ovire(ime_datoteke, ovire):
    f = open(ime_datoteke, "w")
    for y, xs in ovire.items():
        f.write(f"{y:0>3}:")
        for x0, x1 in xs:
            f.write(f"{x0:4}-{x1:<4}")
        f.write("\n")
```

Dodatna naloga

Oddelek za gospodarstvo in motoriziran promet je ponosen na svoje inovacije. (Temu na primer, da so kolesarjem na Slovenski cesti omogočili voziti slalom med avtobusi, z vso resnostjo pravijo "inovacija v svetovnem merilu". Kar dejansko je.)

V okviru te domače naloge so predstavili novo obliko zapisa ovir. Standard MOL-666 zapiše gornje ovire takole:

```
4
5
6
9
11

5
9
11
19
20
30
34

13
5
8
9
11
17
19
22
25
90
100

169
1
2
```

Kot navaja MOL, je novi standard namenjen še večjemu spodbujanju kolesarjenja in trajnostne mobilnosti, hkrati pa tudi hitrejšemu zelenemu prehodu v brezogljično družbo.

Napiši funkcijo `preberi_ovire(ime_datoteke)`, ki prebere ovire iz takšne datoteke v slovar.

Testno datoteko bodo sestavili testi ob prvem zagonu. Njeno ime bo "ovire.txt", vendar naj bo funkcija napisana tako, da zna prebrati datoteko s poljubnim podanim imenom.

Rešitev

Ena pot je tale: preberemo celotno datoteko v en sam niz in jo razbijemo glede na `\n\n`, se pravi glede na prazne vrstice (dva `\n` pač pomenita, da gremo dvakrat zapored v novo vrstico). Z zanko gremo čez te bloke; imenovali jih bomo `vrstica_ovir` (vrstica v smislu vrstice na kolesarski poti, ne v datoteki).

Vrstico ovir razbijemo s `split` na vrstice in vsako z `int` pretvorimo v številko. Tako dobimo seznam številk. Ničta je `y`. Ostale številke so začetki in konci ovir: `stevilke[1::2]` so vsi začetki (začnemo s prvo in jemljemo vsako drugo), `stevilke[2::2]` so konci (začnemo z drugo in jemljemo vsako drugo). Začetke in konce zipnemo skupaj, dobljene pare zložimo v seznam in ga damo v slovar `ovire` pod ustrezen ključ.

```
def preberi_ovire(ime_datoteke):
    ovire = {}
    for vrstica_ovir in open(ime_datoteke).read().split("\n\n"):
        stevilke = [int(x) for x in vrstica_ovir.split()]
        ovire[stevilke[0]] = list(zip(stevilke[1::2], stevilke[2::2]))
    return ovire
```

Druga pot je sprotno branje. Program bo daljši, najbrž tudi počasnejši, je pa zanimiv zato, ker porabi manj pomnilnika, saj bo vedno obdeloval le posamično vrstico. Tu se to sicer ne bo prav nič poznalo, vseeno pa je koristno, da znamo programirati tudi tako.

```
def preberi_ovire(ime_datoteke):
    ovire = {}
    y = None
    f = open(ime_datoteke)
    for vrstica in f:
        if y is None:  # nimam številke vrstice
            y = int(vrstica)
            ovire[y] = []
        elif not vrstica.strip():
            y = None
        else:
            x0 = int(vrstica)
```

```
            x1 = int(f.readline())
            ovire[y].append((x0, x1))
    return ovire
```

y je številka trenutne vrstice. V začetku je ne poznamo, zato bo y enak None. Odpremo datoteko in jo beremo.

- Če je y enak None, v y preberemo številko vrstice. V slovar dodamo novo vrstico.
- Sicer preverimo, ali je vrstica datoteke prazna: v tem primeru je vrstice konec in v y zabeležimo None, da se ve, da čakamo naslednjo vrstico.
- Sicer pa pravkar prebrana vrstica vsebuje začetek ovire. Preberemo (kar z readline) še eno vrstico, to je, konec ovire in to dodamo v seznam, ki ustreza trenutni vrstici.